The use of Enterprise Service Bus to transfer large volumes of data

Tomasz Górski
Faculty of Cybernetics, Military University of Technology, Warsaw, Poland
gorski@wat.edu.pl

Abstract: The paper presents an application of Enterprise Service Bus (ESB) to handle transfer of large volumes of data between IT systems. The use of ESB may affect the reduction of operating costs of the public entity minimizing the number of connections between systems, shortening the time of integration with external systems and increasing the reusability of software written. The design of integration was done according to the ‘1+5’ architectural views model adjusted to the integration solutions description. The previously proposed UML profile UML Profile for Integration Platform has been used. We present the solution, on a specific case study of Agricultural and Food Quality Inspection (AFQI) (www.ijhars.gov.pl), showing how fast data can be transferred electronically using an ESB. Finally, benefits of the solution were analysed in the context of the reduced time of data transfer.

Keywords: Enterprise Service Bus, IT systems integration, ‘1+5’ architectural views model, Service-Oriented Architecture

1. Introduction

Usually, an organization uses multiple IT systems. This forces the need to build integration solutions comprised of IT systems and a communication layer that enables cooperation between these systems [17, 18]. A solution of this type is called an integration platform or an integration solution. This kind of solution is widely realized in Service Oriented Architecture (SOA). SOA was first defined in 1996 by Schulte and Natis [23] as ‘a style of multi-tier computing that helps organizations share logic and data among multiple applications and usage modes’. More complete and robust definition of SOA was presented by Rotem-Gal-Oz in 2012 [20] ‘Service Oriented Architecture (SOA) is an architectural style for building systems based on interactions of loosely coupled, coarse-grained, and autonomous components called services. Each service exposes processes and behavior through contracts, which are composed of messages at discoverable addresses called endpoints. A service’s behavior is governed by polices that are external to the service itself. The contracts and messages are used by external components called service consumers.’ The central element in integration solution is Enterprise Service Bus [9, 17, 20]. When designing integration solutions, it is essential to be able to model their complete architectural description. For this purpose, an architectural views model is needed which allows for modelling integration platforms and a set of model constructs enabling presentation of the integration platform’s entire architecture [13, 15].
The aim of the paper is presentation of the integration solution for Agricultural and Food Quality Inspection (AFQI) (www.ijhars.gov.pl) which automates transferring large volumes of data thanks to application of enterprise service bus. The architectural views model ‘1+5’ presented in [13] is used. The remainder of the paper is structured as outlined below.

Section ‘Architectural views model 1+5’ describes views, models and diagrams of the architectural views model which was used to design integration solution. Section ‘Related studies’ contains an overview of publications dedicated to similar problems, as well as SOA patterns for performance. Thereafter, performance metrics of integration solution were proposed. Subsequently, a case study is discussed. Next, the architecture of the integration solution is presented and followed by the description of functional and non-functional requirements. In addition, the paper presents analysis of performance tests results and discusses benefits of the designed integration solution for the organization. Section ‘Conclusions and further work’ concludes the paper, summing up the subject and outlining directions for further work.

2. Architectural views model ‘1+5’

Consistency of the architectural description of IT solutions is a significant matter and the subject of studies today [1]. A variety of models exists, with differing sets of architectural views, such as, for example: ‘4+1’, RM-ODP, Siemens, SEI views [21]. Yet, they do not allow for a complete description of the integration solutions architecture. The ‘1+5’ model of architectural views proposed here has been accommodated to suit the process of an integration platform design [13]. The following architectural views have been distinguished within the model:

- Integrated Processes,
- Use Cases,
- Logical,
- Integrated Services,
- Contracts,
- Deployment.

Figure 1. The architectural views model ‘1+5’

The view of Integrated Processes is the basic architectural view here. In this view, business processes to be automated on the integration platform are modelled. The next four views (Use Cases, Logical, Integrated Services and Contracts) present the integration platform design. The Use Cases view contains functional requirements for the system being
integrated within the platform. The view of Integrated Services presents services exposed by IT systems, and the way how they are connected to the service bus. The Contracts view shows components representing IT systems and the contracts defined between them. This view also encompasses mediation flows for each contract. The last view, Deployment, shows the way the integration platform elements are deployed in a certain runtime environment. Figure 1 illustrates the architectural views model ‘1+5’.

Table 1. Elements for modelling the integration platform architecture

<table>
<thead>
<tr>
<th>Model</th>
<th>View</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processes</td>
<td>Integrated Processes</td>
<td>(BPMN) Business Process</td>
</tr>
<tr>
<td>Use Cases</td>
<td>Use Cases</td>
<td>(UML) Use Case</td>
</tr>
<tr>
<td>Design</td>
<td>Logical</td>
<td>(UML) Sequence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(UML) Communication</td>
</tr>
<tr>
<td>Services</td>
<td>Integrated Services</td>
<td>(UML) Component</td>
</tr>
<tr>
<td></td>
<td>Contracts</td>
<td>(UML) Component</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(UML) Activity</td>
</tr>
<tr>
<td>Deployment</td>
<td>Deployment</td>
<td>(UML) Composite Structure</td>
</tr>
</tbody>
</table>

A detailed description of the architectural views model referred to above and examples of its application can be found in the literature [13, 14]. In the approach analysed here, models and diagrams of BPMN [5, 7] and UML [16, 25] languages, with an extension of Service oriented architecture Modeling Language stereotypes [24] have been used for modelling the integration platform architecture (Table 1).

3. Related studies

Service-oriented architecture [9] and IT systems integration with application of Enterprise Service Bus are current fields of research and the literature on the subject is rich in integration problems. A systematic literature review on the subject of process models for service based applications was given by Lane [22]. In publications of recent years, special focus has been on the topical and vital issues of business process support by IT systems. Business processes support through IT systems in the context of optimal selection of services to carry out tasks in the business process is discussed by Ardagna [3]. Furthermore, subject of modelling functional requirements for configurable content- and context-aware dynamic service selection in business process models is raised by Frece [11]. Another vital problem, in the field of service-oriented architecture, is IT support for dynamically reconfigurable processes which is presented by Cicirelli [8]. Optimization of integration flows [4] and semi-automated software service automation [2] are very important problems discussed in the literature of the subject in the recent time. In addition to functional requirements, non-functional requirements play a key role in designing integration solutions. Details of the solution adaptation in service-oriented architecture taking into account cost, reliability, availability and performance requirements is presented by Potena [19]. As far as performance is concerned, Górski presented performance analysis of selected frameworks for integration solutions development [12]. Integration of many various IT systems makes the integration project highly complex [6]. Most commonly, an organization operates a variety
of IT systems. This implies the need to build integration solutions comprised of IT systems and a communication layer that enables cooperation between these systems [18]. In addition, in the literature of the subject, there are present works describing elements for building new generation of integrated business environments with analysis of Enterprise Service Bus features as the core of this infrastructure [17]. An up-to-date, comprehensive comparison of the characteristics of enterprise bus service from different vendors is given by Górski [14]. With the maturing of SOA, in the integration solutions there were identified architectural patterns [20]. And last but not least, in the construction of integration solutions an important subject is reliable delivery of the correct message. A proposal of errors detection in Enterprise Application Integration solutions is given by Frantz [10]. One very important non-functional requirement type is performance. Many people believe that performance is strictly connected with hardware. But, there is a lot to gain with proper software architecture in terms of architectural patterns.

The following architectural patterns will be discussed [20]:

- **Decoupled Invocation** – handle normal request loads, peak request loads, and continuous periods of time at high load without failing,
- **Parallel Pipelines** – build services that maintain state and high throughput,
- **Gridable Service** – build services to handle computationally intense tasks in a scalable manner,
- **Service Instance** – build services that are scalable in a simple and cost-effective way,
- **Virtual Endpoint** – provide services with location transparency that gracefully recover from failure without affecting service consumers,
- **Service Watchdog** – increase availability and identify and resolve problems and failures that are service-specific.

The Decoupled Invocation pattern helps to handle peak loads by queuing up requests. But this solution doesn’t increase overall service scalability when request rates increase. Under a continuous high-request rate, the requests can accumulate in the queue and eventually overflow. The Parallel Pipelines pattern is devoted to increase overall service throughput. For the computationally intensive subtasks within a pipeline the Gridable Service pattern may be applied. Often, services need to be distributed to help with computationally intensive tasks. The Service Instance pattern allows on deployment multiple copies of the service. The Decoupled Invocation pattern is related to the Service Instance pattern. To combine the two, you implement the service instances as multiple readers that process the same input queue. The Virtual Endpoint pattern provides location transparency, which provides availability and scalability. The Service Watchdog pattern shows how a service can proactively identify faults and try to heal itself when it finds problems.

### 4. Performance metrics of integration solution

One of the basic performance metrics is time of getting the results of the service execution. You can take the following performance measures of an integration platform:

- **One-Way Delay (OWD)** – this is the time it takes the message to reach a target location, which is calculated from the moment of sending message from the source point to reach the service itself. The OWD consists of two elements that are associated with the transmission of data: propagation delay (the time to simply transport a message from source to target), serialization delay (the time it takes to mediate the currently processed message).
- Round Trip Time (RTT) – this time is measured from the moment of sending a message by requester to receive a reply by the requester. It would seem that it is just twice the OWD time, but in reality may be quite different times.
- Maximum delay – the highest value of delay in one direction. It is important for applications that have established the maximum deviation of the delay time.

For asynchronous calls the OWD can be applied. For this type of call typical measure is the value of the number of messages handled in a given period. In case of synchronous calls the RTT is suitable for use.

5. Case study

In the paper, part of functioning of Agricultural and Food Quality Inspection (AFQI) is considered. Agricultural and Food Quality Inspection (AFQI) started its activity on 1 January 2003 under the national Act of December 21, 2000 on Commercial Quality of Agricultural and Food Products. Agricultural and Food Quality Inspection is the competent authority supervising quality of agricultural and food products in Poland. In the framework of this broad task, AFQI carries out quality controls at production, domestic market, as well as import and export stage. One of the tasks of the Inspection is also supervising the certification bodies and organic farming. The certification bodies are entities authorized by the Minister of Agriculture and Rural Development to carry out inspections and issuing and revoking certificates in the field of organic farming. They collect data from organic producers. One of the duties of certifying bodies is also to provide The Agency for Restructuring and Modernisation of Agriculture (ARMA) the list of organic producers who have met certain requirements in organic farming. ARMA as an accredited paying agency involved in the implementation of instruments co-financed from the EU budget and assist with national funds. ARMA is responsible for providing financial assistance to organic producers implementing a package of organic farming within the Rural Development Programme, financed by the European Fund for Rural Development. Payments are granting based on the data contained in the list notified by the certification bodies. A separate list of organic producers is transmitted by the certification bodies to AFQI. ARMA periodically carry out inspections of organic producers aimed at checking the implementation of agricultural and environmental commitments and, consequently, the correct preparation of the list.

In case of irregularities in the list, the President of ARMA submit request to AFQI to punish the certification body. Such an application often contains multiple attachments (e.g.: photos, documents) on agricultural plots that are transferred on CD/DVD disc. Such files concerning one case of irregularity may have a size of about 700 MB. Figure 2 shows a scheme showing the flow of data. Due to the large volume of data unfortunately it is not possible to transfer data by ePUAP, because it prevents the transfer of such large attachments.

![Figure 2. Data flow schematically of the organic farming system in Poland](image-url)
Regardless of the list transmitted to the President of ARMA, certification bodies shall transmit the list to AFQI as well as data and information about organic producers. This data is used to supervise the certification bodies and organic farming. The certification bodies provide data offline to AFQI in an XML file format (e.g. on CD/DVD disc), which are then manually imported into AFQI Integrated Information System (AFQI IIS).

6. Requirements and architecture of the integration solution

With this in mind, and having AFQI prospect of integration with the Customs Service and a ePUAP platform, deployment of enterprise service bus would appear to be the best solution. For example, an event in ARMA system could cause execution of process instance in the AFQI IIS associated with this event realization. AFQI IIS is a key AFQI information system enabling electronic registration of tasks performed by the Inspection. The system consists of the following subsystems: registration of inspections, registration of contractors, inspectors, security, dictionaries, front-end, data analysis.

Eventually, the system has to cover all the tasks carried out by the Inspection and to allow easy integration with external systems. Therefore, it is necessary to isolate additional communication subsystem whose function should be to handle large volumes of data. The figure (Figure 3) shows the architecture of the integration solution including enterprise service bus as a part responsible for IT systems integration. In component diagram there were used: stereotype <<ESB>> for enterprise service bus (from UML Profile for Integration Platform [15]) and stereotypes from SoaML language: <<Capability>>, <<Consumer>>, <<Provider>>. There are possible following models of communication: request / response, request / multi-response, event propagation and publish / subscribe. In the solution, the first model seems to be appropriate. It is assumed that the consumer message is sent to the target service, processed there, and then the answer is returned. It should be noted that this model can be realized both in synchronous and asynchronous modes.

Figure 3. UML component diagram of the integration solution architecture
In terms of handling requests sent between IT systems there can be distinguished the following mediation patterns [14]: protocol switch, transform, enrich, route, distribute, monitor, correlate. In the integration solution, transform pattern is used which ensures transformation of the message from object format served by enterprise service bus to XML format.

The main function to test the integration solution is use case ‘Send request to penalize’ (Figure 4). Thanks to this function request to penalize specific certification body can be sent from ARMA IT system to AFQI IIS electronically. The stereotype <<IntegratedSystem>> from UML Profile for Integration Platform [15] was used in the figure. This stereotype was used to emphasize the fact that AFQI IIS is connected through integration platform. Deployment of such solution in AFQI and ARMA would replace sending the paper copy of request with the annexes on the CD/DVD.

![Figure 4. Use case diagram of the integration solution.](image)

Furthermore, the integration solution should meet the following requirements: provide a uniform interface to transmit and receive data, allow easy addition of data filtering, automate the data transmission, provide reliable message delivery, ensure authorization of the sender and the recipient, realize verification of XML documents for compliance with XSD schema, ensure processing each message only once.

7. Performance tests analysis

Tests were carried out for the use case ‘Send request to penalize’. As part of documentation of the plot there are the following documents: cover letter, a sketch of the plot (PDF) and plot pictures (jpg). A total size of documentation to be sent is about 700 MB. The message contained *.pdf files and *.jpg files encoded by base64 algorithm. SOAP Message Transmission Optimization Mechanism (MTOM) is a standard developed by W3C to ensure the efficient transmission of binary data using web services. Using the XML Optimized Packaging (XOP) mechanism data is sent as a SOAP message MIME attachments. Reference to the data in a SOAP message is realized by the <xop: Include>. Thanks to this, a text data is separated from a binary data contained in a message.

Test No. 1 involved sending and receiving one message containing all photos and sketches of plots for the selected case. Size of documentation for testing was 675.71 MB. After about 10 minutes from test script launch there were a series of warnings and the script aborted its operation. Despite increasing memory allocation limit for the test script the message was not delivered. This leads to a conclusion, that sending a single message that large, containing all files, is not recommended.

Test No. 2 consisted of sending each file as a separate message. Size of documentation for this test was also 675.71 MB. On the other hand, the number of all files was 169, and each file was sent as a separate message. Results of the test contains table 1.
The use of Enterprise Service Bus to transfer large volumes of data

Table 1. Results of test No. 2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>127,86</td>
<td>1,32</td>
<td>4</td>
<td>6,11</td>
<td>0,15</td>
<td>0,49</td>
<td>1,31</td>
</tr>
</tbody>
</table>

Results of test No. 1 and test No. 2 clearly show the importance of appropriate choice of method of sending documentation using enterprise service bus.

Moreover, test No. 3 was conducted. Test No. 3 consisted of sending messages, each about the size of 4.2 MB with the increasing number of simultaneously transmitted messages. The purpose of the test was to show impact of the number of transmitted messages on the time of transmission. Test No. 3 was performed using JMeter. Results of the test contains table 2.

Table 2. Results of test No. 3

<table>
<thead>
<tr>
<th>Number of messages</th>
<th>Minimum time of sending single message [s]</th>
<th>Mean time of sending single message [s]</th>
<th>Maximum time of sending single message [s]</th>
<th>Size of sent messages [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4,63</td>
<td>5,78</td>
<td>6,10</td>
<td>42</td>
</tr>
<tr>
<td>50</td>
<td>25,73</td>
<td>36,08</td>
<td>45,62</td>
<td>210</td>
</tr>
<tr>
<td>100</td>
<td>24,83</td>
<td>62,40</td>
<td>92,54</td>
<td>420</td>
</tr>
<tr>
<td>200</td>
<td>25,10</td>
<td>108,66</td>
<td>185,02</td>
<td>840</td>
</tr>
<tr>
<td>500</td>
<td>26,62</td>
<td>178,45</td>
<td>310,40</td>
<td>2100</td>
</tr>
</tbody>
</table>

In addition, the figure (Figure 5) shows a chart of mean time of sending a single message depending on the number of messages sent simultaneously through the enterprise service bus.

![Figure 5. Chart of mean time of sending a single message](image-url)
Tests have shown that the functional requirement consisting of sending data using the enterprise bus services has been met. A partial implementation of the architectural pattern Decoupled Invocation has increased the performance and allowed on effectively dealing with a short-term heavy load. In the present case this is important because data is sent seldom, but in a relatively short period of time may cause heavy load.

8. Conclusions and further work

The paper presents an application of Enterprise Service Bus (ESB) to handle transfer of large volumes of data between IT systems. The use of ESB may affect the reduction of operating costs of the public entity minimizing the number of connections between systems, shortening the time of integration with external systems and increasing the reusability of software written. The design of integration was done according to the ‘1+5’ architectural views model adjusted to the integration solutions description. The previously proposed UML profile UML Profile for Integration Platform has been used. The use of ‘1+5’ architectural views model has helped to present the integration solution architecture in an intelligible form.

We present the solution, on a specific case study of Agricultural and Food Quality Inspection (AFQI), showing how fast data can be transferred electronically using an ESB. It should be emphasized that the data transfer via postal mail takes an average of 3 days, and sending an electronic version for an exemplary case took about 2 minutes. As you can see the potential benefits of implementing this type of solution are enormous. Results of test No. 1 and test No. 2 clearly show the importance of appropriate choice of method of sending documentation using enterprise service bus. A partial implementation of the architectural pattern Decoupled Invocation has increased the performance and allowed on effectively dealing with a short-term heavy load.

Further work will focus on the implementation of architectural patterns in order to increase the performance of the integration solution. In addition, further work will be aimed at implementing production version of the integration solution in the Agricultural and Food Quality Inspection (AFQI).

References

The use of Enterprise Service Bus to transfer large volumes of data